

**18CSC302J- COMPUTER NETWORKS  
MINOR PROJECT REPORT**

*Submitted by*

**Praneet Mishra (RA2011051010069)**

*Under the Guidance of*

***Dr. Jeba Sonia***

**Associate Professor, Department of Networking and Communications**

*In partial satisfaction of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY in COMPUTER  
SCIENCE ENGINEERING**

**with specialisation in Gaming Technology**



**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

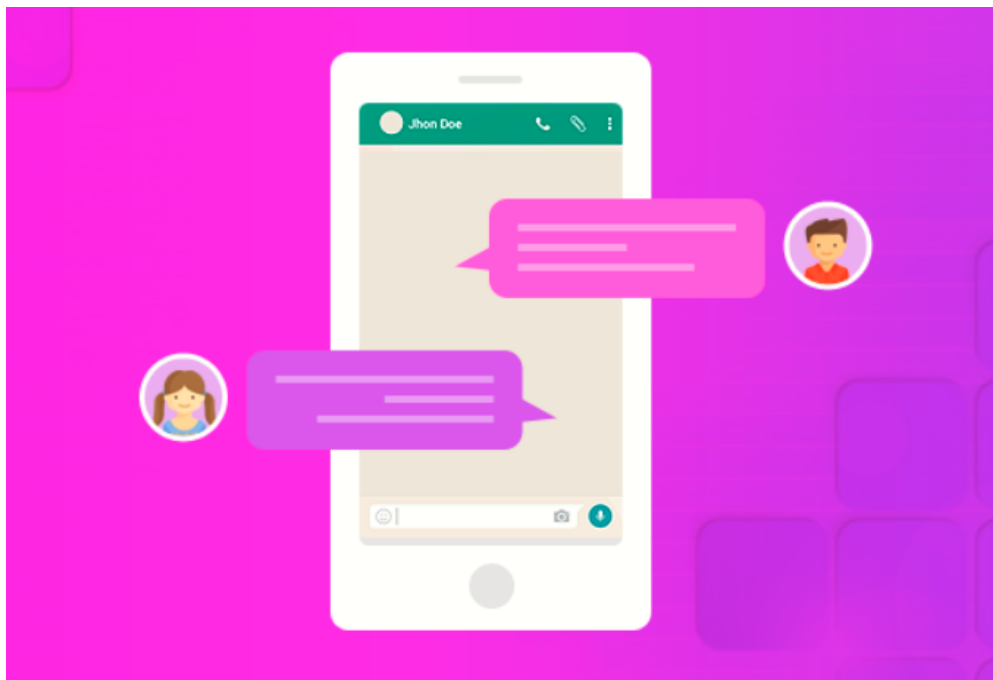
**KATTANKULATHUR - 603203**

**November 2022**

# **FTP CLIENT SERVER MODEL (MULTIMEDIA)**

## **Aim**

To Implement multimedia chat application using FTP and full Duplex for chatting, sending photos and text file with friends by using C language.



## **MULTIMEDIA CHAT APPLICATION**

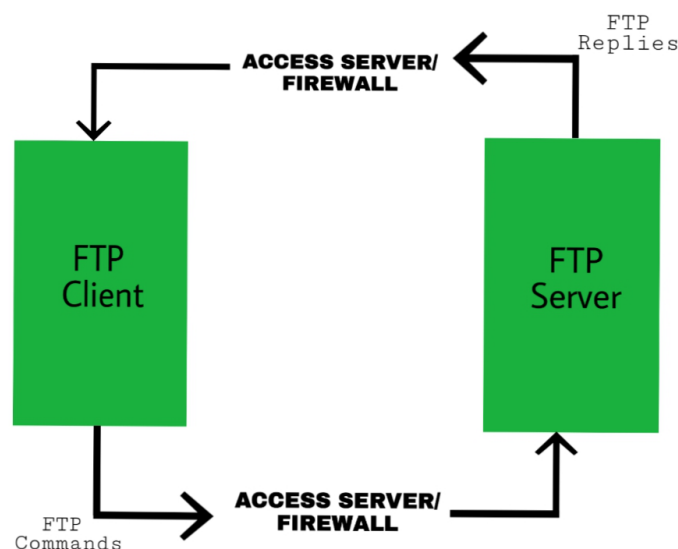
## Introduction

In view of, a modern chat app must possess noteworthy functionalities which can be adaptable to any type of chat solutions. In recent times, it has been found that creating chat app by using C language. An ideal chat app has the potential of offering solutions to send files, talk without interruption and can also send photos by using C language.

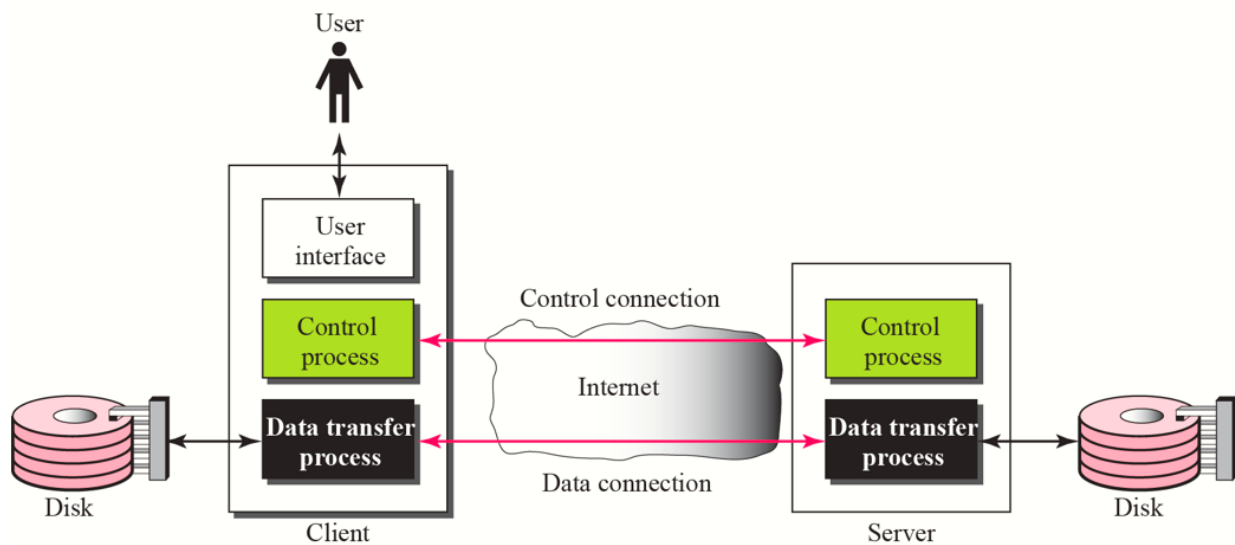
The FTP connection is established between two systems and they communicate with each other using a network. So, for the connection, the user can get permission by providing the credentials to the FTP server or can use anonymous FTP.

When an FTP connection is established, there are two types of communication channels are also established and they are known as command channel and data channel. The command channel is used to transfer the commands and responses from client to server and server to client. FTP uses the same approach as TELNET or SMTP to communicate across the control connection. It uses the NVT ASCII character set for communication. It uses port number 21. Whereas the data channel is used to actually transfer the data between client and server. It uses port number 20.

The FTP client using the URL gives the FTP command along with the FTP server address. As soon as the server and the client get connected to the network, the user logs in using User ID and password. If the user is not registered with the server, then also he/she can access the files by using the anonymous login where the password is the client's email address. The server verifies the user login and allows the client to access the files. The client transfers the desired files and exits the connection. The figure below shows the working of FTP.



In the below figure you can see the basic functions done by Client-Server programs using FTP protocol for establishment of a successful connection, communication and finally connection teardown or termination.

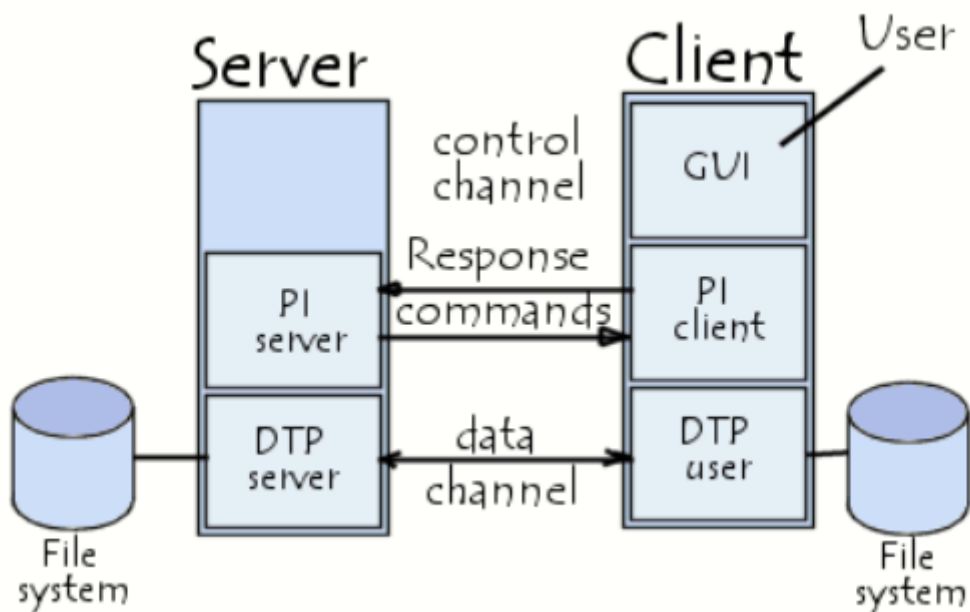


In this configuration, the protocol imposes that the control channels remain open throughout the data transfer. So a server can stop transmission if the control channel is broken during transmission.

During an FTP connection, two transmission channels are open:

A channel for commands (control channel).

A channel for data.



## **Objectives**

Objectives of Multimedia chat project are given below :

- Create a Multimedia Chat using FTP protocol to chat with friends, send photos and also sends text files.
- By creating chat application using C program gives good hand on it.
- Gives a great knowledge in Computer Networking by using FTP protocol.

## **Description**

### **Socket programming**

Sockets can be thought of as endpoints in a communication channel that is bi-directional and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server-side. Any client that has a socket associated with the same port can communicate with the server socket.

### **FTP Server**

The primary purpose of an FTP server is to allow users to upload and download files. An FTP server is a computer that has a file transfer protocol (FTP) address and is dedicated to receiving an FTP connection. FTP is a protocol used to transfer files via the internet between a server (sender) and a client (receiver). An FTP server is a computer that offers files available for download via an FTP protocol, and it is a common solution used to facilitate remote data sharing between computers.

### **Full Duplex Communication**

Full-duplex data transmission means that data can be transmitted in both directions on a signal carrier at the same time. For example, on a local area network with a technology that has full-duplex transmission, one workstation can be sending data on the line while another workstation is receiving data. Full-duplex transmission implies a bidirectional line that can move data in both directions simultaneously.

## Implementation

### Code

#### Chat Server

/\* Implementing C language code for Full Duplex \*/

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>

int main(int argc,char *argv[])
{
int clientSocketDescriptor,socketDescriptor;

struct sockaddr_in serverAddress,clientAddress;
socklen_t clientLength;

char recvBuffer[1000],sendBuffer[1000];
pid_t cpid;
bzero(&serverAddress,sizeof(serverAddress));

serverAddress.sin_family=AF_INET;
serverAddress.sin_addr.s_addr=htonl(INADDR_ANY);
serverAddress.sin_port=htons(5500);

socketDescriptor=socket(AF_INET,SOCK_STREAM,0);

bind(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));

listen(socketDescriptor,5);
printf("%s\n","Server is running ...");

clientSocketDescriptor=accept(socketDescriptor,(struct
sockaddr*)&clientAddress,&clientLength);

cpid=fork();

if(cpid==0)
{
while(1)
```

```

{
bzero(&recvBuffer,sizeof(recvBuffer));

recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nCLIENT : %s\n",recvBuffer);
}
}
else
{
while(1)
{

bzero(&sendBuffer,sizeof(sendBuffer));
printf("\nType a message here ... ");

fgets(sendBuffer,10000,stdin);

send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
printf("\nMessage sent !\n");
}
}
return 0;
}

/* Successfully implemented multimedia chat application by using FTP */

#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5033
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
    int sockfd, newsockfd, clength;
    struct sockaddr_in serv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");

```

```

bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
printf("\nListening...");
listen(sockfd, 5);
clength=sizeof(cli_addr);
newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);
close(sockfd);
read(newsockfd, &str, MAX);
printf("\nClient message\n File Name : %s\n", str);
f1=fopen(str, "r");
while(fgets(buff, 4096, f1)!=NULL)
{
    write(newsockfd, buff,MAX);
    printf("\n");
}
fclose(f1);
printf("\nFile Transferred\n");
return 0;
}

```

### **Chat Client**

/\* Implementing C language code for Full Duplex \*/

```
#include "stdio.h"
```

```
#include "stdlib.h"
```

```
#include "string.h"
```

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
```

```
#include <unistd.h>
```

```
#include "netdb.h"
```

```
#include "arpa/inet.h"
```

```
int main()
```

```
{
```

```
int socketDescriptor;
```

```
struct sockaddr_in serverAddress;
```

```
char sendBuffer[1000],recvBuffer[1000];
```

```
pid_t cpid;
```



```
bzero(&serverAddress,sizeof(serverAddress));

serverAddress.sin_family=AF_INET;

serverAddress.sin_addr.s_addr=inet_addr("127.0.0.1");

serverAddress.sin_port=htons(5500);

socketDescriptor=socket(AF_INET,SOCK_STREAM,0);

connect(socketDescriptor,(struct

sockaddr*)&serverAddress,sizeof(serverAddress));

cpid=fork();

if(cpid==0)

{

while(1)

{

bzero(&sendBuffer,sizeof(sendBuffer));

printf("\nType a message here ... ");

fgets(sendBuffer,10000,stdin);

send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);

printf("\nMessage sent !\n");

}

}

else

{

while(1)

{

bzero(&recvBuffer,sizeof(recvBuffer));

recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);

printf("\nSERVER : %s\n",recvBuffer);
```

```
}  
  
}  
  
return 0;  
  
}
```

**/\* Successfully implemented multimedia chat  
application by using FTP \*/**

```
#include<stdio.h>  
#include<sys/types.h>  
#include<sys/socket.h>  
#include<netinet/in.h>  
#include<netdb.h>  
#include<stdlib.h>  
#include<string.h>  
#include<unistd.h>  
#define SERV_TCP_PORT 5033  
#define MAX 60  
int main(int arg,char*argv[])  
{  
    int sockfd,n;  
    struct sockaddr_in serv_addr;
```

```

struct hostent*server;

char send[MAX],recvline[MAX],s[MAX],name[MAX];

sockfd=socket(AF_INET,SOCK_STREAM,0);

serv_addr.sin_family=AF_INET;

serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

serv_addr.sin_port=htons(SERV_TCP_PORT);

connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));

printf("\nEnter the source file name : \n");

scanf("%s",send);

write(sockfd,send,MAX);

while((n=read(sockfd,recvline,MAX))!=0)

{printf("%s",recvline); }

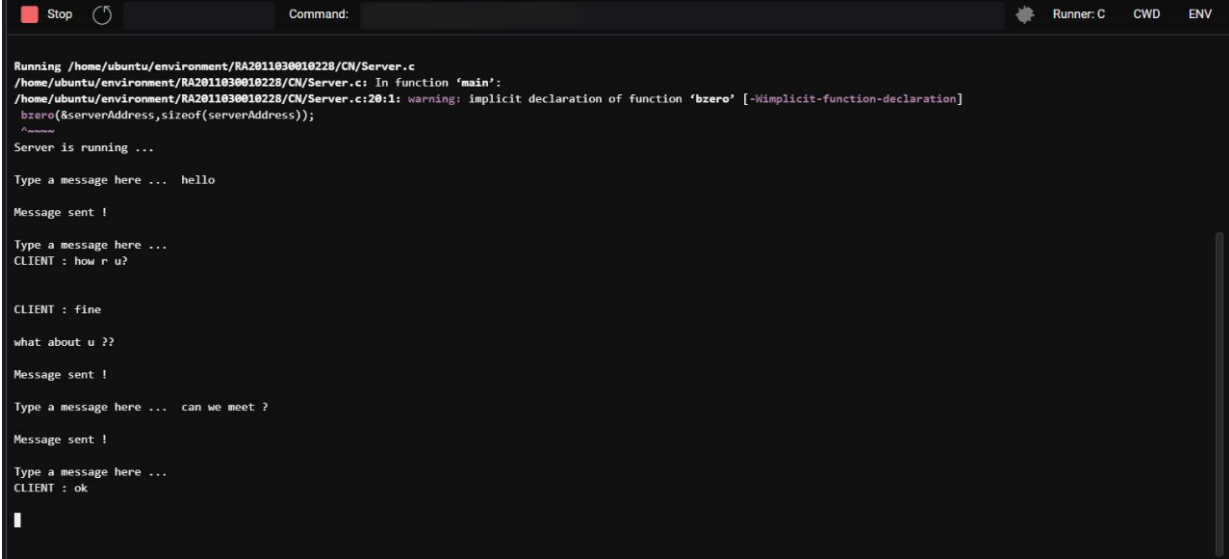
close(sockfd);

return 0;

}

```

## Output



```
Running /home/ubuntu/environment/RA2011030010228/CN/Server.c
/home/ubuntu/environment/RA2011030010228/CN/Server.c: In function 'main':
/home/ubuntu/environment/RA2011030010228/CN/Server.c:20:1: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
  bzero(&serverAddress,sizeof(serverAddress));
  ^~~~~~
Server is running ...

Type a message here ...  hello

Message sent !

Type a message here ...
CLIENT : how r u?

CLIENT : fine

what about u ??

Message sent !

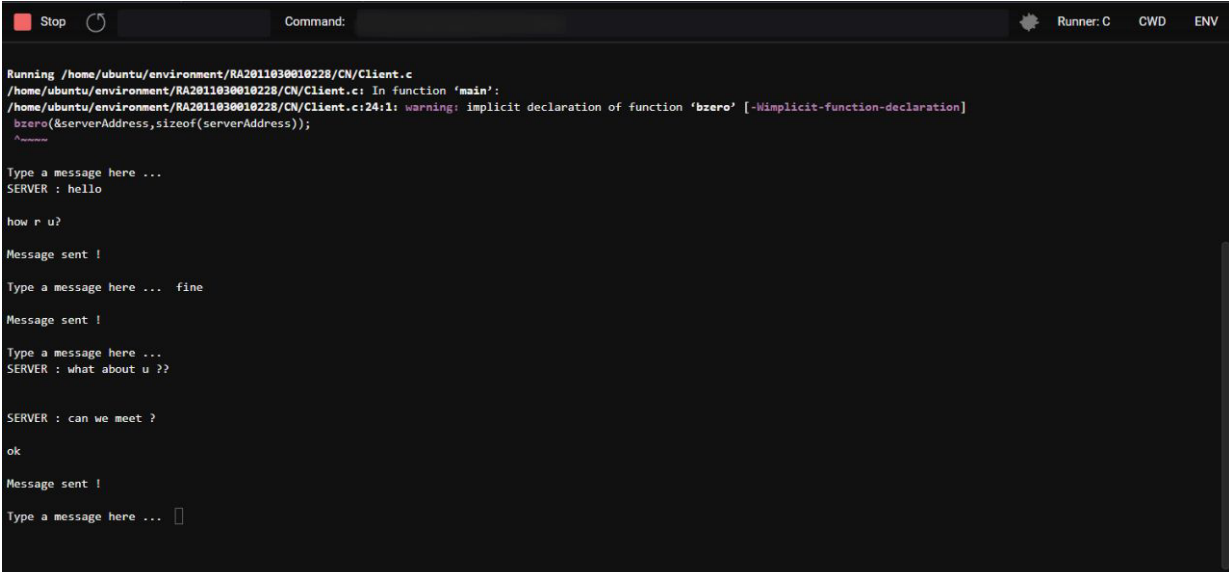
Type a message here ...  can we meet ?

Message sent !

Type a message here ...
CLIENT : ok

█
```

## Server for Full Duplex



```
Running /home/ubuntu/environment/RA2011030010228/CN/Client.c
/home/ubuntu/environment/RA2011030010228/CN/Client.c: In function 'main':
/home/ubuntu/environment/RA2011030010228/CN/Client.c:24:1: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
  bzero(&serverAddress,sizeof(serverAddress));
  ^~~~~~

Type a message here ...
SERVER : hello

how r u?

Message sent !

Type a message here ...  fine

Message sent !

Type a message here ...
SERVER : what about u ??

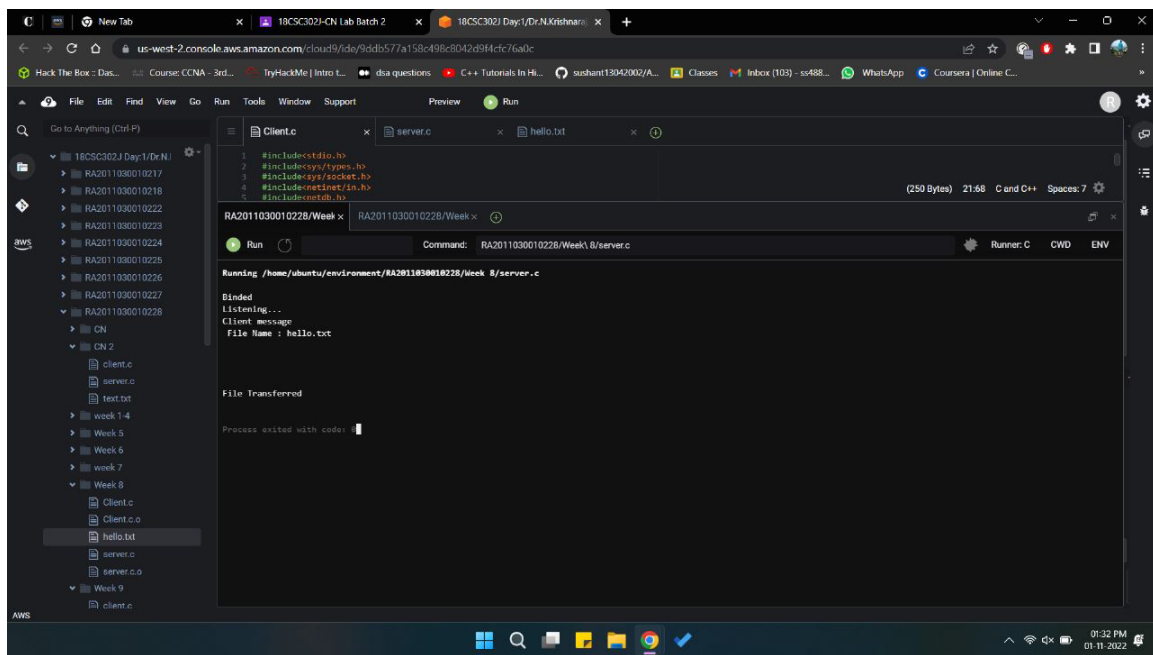
SERVER : can we meet ?

ok

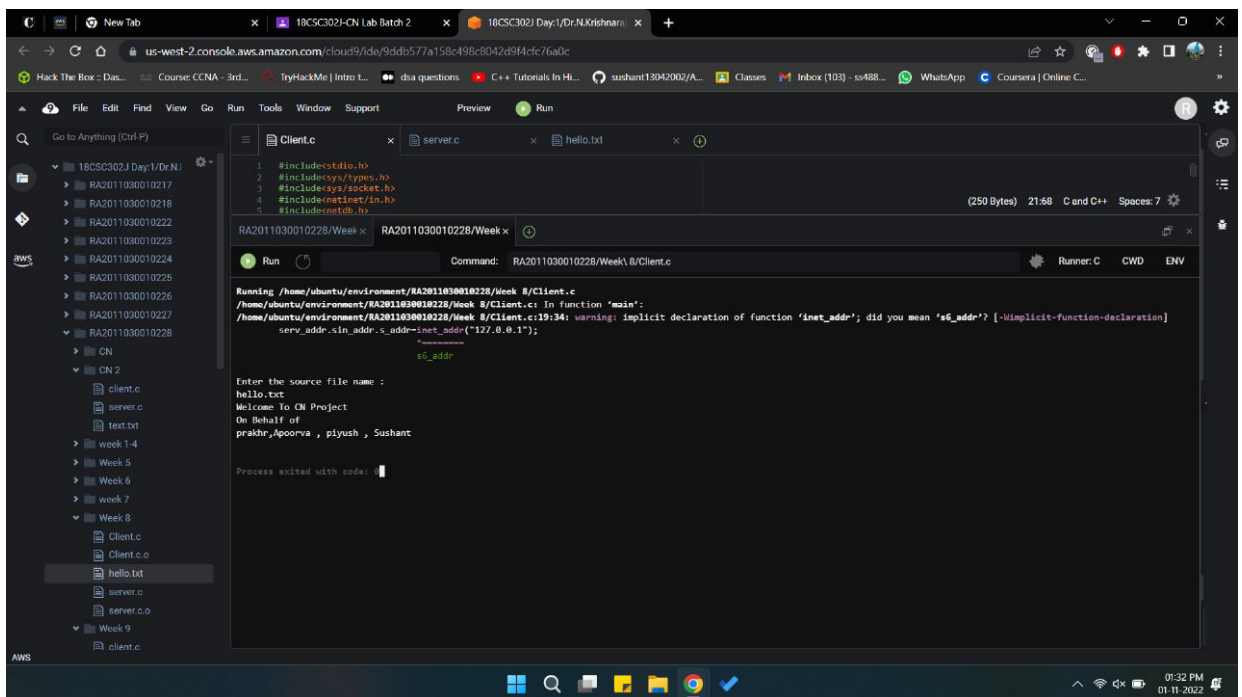
Message sent !

Type a message here ... █
```

## Client for Full Duplex



## Server for FTP



## Client for FTP

## **Result :**

FTP were used to implement Full Duplex Communication between Client and Server to transfer files, text, data and same was demonstrated in Amazon Web Services.

### References / Websites

1. FTP MODEL IN NESO ACADEMY
2. FTP MODEL in GeeksforGeeks
3. Multimedia chat working in getstream.io
4. Full Duplex on javatpoint
5. Client Server for FTP on [www.technopedia.com](http://www.technopedia.com)